APPENDIX 1

White Paper: InfoSeer Audio Scan Techniques

This paper is intended to summarize the capabilities of the audio scan technique developed at InfoSeer and provide a description of the algorithm.

The audio scan technology relies on two proprietary algorithms:

- Scan Data Production – Used to produce a tag data structure for a given audio source
- Scan Data Compare – Used to compare two tag data structures and produce a 'percent match' value

Scan Capabilities

The scan algorithm provides the following functional features:

- Level Shift Insensitive – If the same source is presented at two different volume levels, it should be recognized as such (equal).
- Stereo 'Balance' Insensitivity – Stereo sources are recognized independent of the direction (left and / or right channel) of the source data.
- Ignore Leading 'Quiet' Data – This feature waits for the input level to exceed a fixed value before actual processing begins. (The fixed threshold is very low and is intended primarily to ignore blocks of leading samples that are near zero level. It is likely that these blocks are artifacts produced by the software used to store the original data.)

- Time Shifting Insensitivity – If someone were to remove the first n seconds from a song we can still recognize that song as long as n is less than around 5.0 seconds.

- Time Compression Insensitivity – Radio stations sometimes transmit time compressed audio so that they can have more time for commercials. I'm guessing the industry standard is around 15% compression (85% of the original). In limited testing it was determined that we could support this by producing a scan of the compressed source using a section size that is 85% of the original (e.g., if the uncompressed original is scanned using a 30.0 second section size, a scan of the 15% compressed version with a 25.5 second section time will match the original).

- 'Whole Source' Option – When this is enabled; the available source is scanned once to determine its length in time. Then the section time is computed using the specified number of sections (section time = (whole source time – leading quiet time) / number of sections) so that when a second pass is made the whole source (minus the leading quiet data) is used to compute a tag. This option is appropriate for the case where the source is available in its entirety (e.g., local file or URL, not a streaming source) and a higher degree of recognition is desirable and possible (e.g., InfoWatch).

Scan Data Production Parameters

We developed a flexible audio scanning algorithm that allows us to choose the following parameters for the scan:

- Section Time – Amount of source (in time) to use for scanning for each section. This is a real number greater than zero.
- Number Of Sections – Number of source sections to use when computing the scan data. This is an integer greater than zero.
- Points Per Section – Number of scan data points to produce for each section. Integer greater than zero.

We currently use 30.0 seconds, 1 and 24 for these values in InfoMart.

Scan Production Algorithm

The algorithm operates on 16 bit audio samples (stereo or mono, knowledge of the associated sample rate is required). A FFT (Fast Fourier Transform) size is selected to maintain a desired bin size[1] in the output based on the sample rate.

The input data is down sampled (if possible) then filtered through a low pass filter. This removes noise and other interferences that could affect the accuracy of the result. Also there is statistically little audio data at the higher frequencies. The data is processed with the FFT and the output magnitude data is accumulated in a result vector. Prior to the FFT, a weighting window is applied to the input data. FFT operations can be optionally

[1] 2.691650 Hz / bin, selected for performance reasons based on common sample rate of 44100 Hz for commercial audio. Under certain circumstances a DCT (Discreet Cosine Transform) may be used separately or in addition to the FFT and the results could be summed.

overlapped on the input data by 50% if desired. When all input samples have been processed the section is complete.

This process is repeated for all desired scan sections, producing a separate result vector for each section. Each section result vector is then normalized based on the peak magnitude value over all sections. The specified number of points with the highest magnitude are then selected for each section. Each selected point is stored as a magnitude and frequency pair.

At this point the data is ready for storage or comparison with other scan data.

Scan Compare Parameters

We developed a flexible audio scanning algorithm that allows us to choose the following parameters for the scan:

- Frequency Weight – Amount of "importance" (from 0.0 to 1.0) applied to the frequency value when comparing data points.
- Magnitude Weight – Amount of "importance" (from 0.0 to 1.0) applied to the magnitude value when comparing data points.
- "Fast Track" Ellipse Magnitude – This value is computed from a fixed magnitude and frequency pair that has had the weights described above applied to each associated component. The value is used in a threshold test as described below.

Scan Compare Algorithm

The primary task of the compare algorithm is to compare the two sets of scan data points (referred in the following as scan A and B) created by the scan production algorithm and produce a 'percent match' result.

The first pass of the compare algorithm is to step through each point of scan A (within each section) and find the closest point in scan B using a two dimensional linear distance based on magnitude and frequency. Since there are many more data points available than are needed to achieve a high confidence level for the match, only the closest and high level points are used in the process. This technique further improves the robustness of the detection system.

The influence of each dimensional component (magnitude and frequency) on the distance calculation can be adjusted using weighting values between 0 and 1. This associates a level of 'importance' when comparing of either the magnitude or frequency when comparing data points. The distance values for each point in A is stored in an output array.

Any point in B that was not selected at least once by a point in A (as being closest), is also compared with each value in A to find the minimum distance and stored in the array. Processing then continues on the output array. If a specified percentage of the values in the output array are below a fixed threshold, these values are used in the final 'percent match' computation. Otherwise, the entire output array is used in the final computation. For the percent match, the average distance within each section and across all sections is used in the following equation:

PercentMatch = 100.0 - AverageDistance * MatchScale

This equation will produce negative percent match values that are often limited to 0% for display to the user.

The MatchScale constant is used to adjust how "quickly" the percent match will fall away from 100%. In our system we use a value greater than 95% to indicate a positive match.